Appendix A:

Exemplary HTC File Defining An Exemplary CTS Resource Tag

```
<!--
Component:
    resource element behavior

Applies to:

Usage:
    <html xmlns:crs="www.microsoft.com/hotmail/hmuidev/crs">
    <?IMPORT namespace="crs" implementation="resource.htc" >
    <crs:resource resource-id="59"/>
-->


<PUBLIC:COMPONENT tagname="resource" literalContent="false" lightWeight="true">
    <PUBLIC:METHOD name="refresh" />
    <PUBLIC:PROPERTY NAME="resource-id" ID="propid" GET="get_rid" PUT="put_rid" />
    <PUBLIC:ATTACH event="oncontentready" onevent="contentReady()" />
    <SCRIPT language="JScript">
            var m_rid = null;
            function get_rid()
            {
                    return m_rid;
            }
            function put_rid(new_rid)
            {
                    m_rid = new_rid;
            }
            function contentReady()
            {
                    refresh();
            }
            function refresh()
            {
                    //validate member
                    var html = "";
                    if (m_rid)
                    {
                            if (isFinite(m_rid) == true)
                            {
                                    html = context.getCompiledResourceValue(m_rid, element);
                            }
                            else
                            {
```

MS1-1483US

```
                    html = "(error: invalid resource id: " + m_rid + ". )";
            }
    }
    //update element
    if (element.innerHTML != html)
    {
            try
            {
                    element.innerHTML = html;
            }
            catch(e)
            {
                    //unable to set html into element
            }
    }
}
</SCRIPT>
</PUBLIC:COMPONENT>
```

)

Appendix B:

Exemplary HTC File Defining An Exemplary CTS If-Resource Tag

```
<!--
Component:
    conditional element behavior

-->
<PUBLIC:COMPONENT tagname="if-resource" literalContent="nested" lightWeight="true">
    <PUBLIC:METHOD name="refresh" />
    <PUBLIC:PROPERTY NAME="resource-id" ID="propid" GET="get_rid" PUT="set_rid" />
    <PUBLIC:ATTACH event="oncontentready" onevent="contentReady()" />
    <SCRIPT language="JScript">
            var m_rid = null;
            var m_oadj = null;
            function init()
            {
                    m_oadj = window.document.createElement("span");
                    element.insertAdjacentElement("afterEnd", m_oadj);
            }
            function get_rid()
            {
                    return m_rid;
            }
            function set_rid(new_rid)
            {
                    m_rid = new_rid;
            }
            function contentReady()
            {
                    //content is ready to be rendered
                    init();
                    refresh();
            }
            function refresh()
            {
                    var render = false;

                    if (m_rid)
                    {
                            if (isFinite(m_rid) == true)
                            {
                                    //determine if there is an instance node which meets the criteria
                                    var c_html = context.getCompiledResourceValue(get_rid());
                                    if (c_html)
```

```
                        {
                                if (c_html.toString().length > 0)
                                {
                                        render = true;
                                }
                        }
                }
        }
        if (render == true)
        {
                m_oadj.innerHTML = element.innerHTML;
        }
        else
        {
                m_oadj.innerHTML = "";
        }
}
</SCRIPT>
</PUBLIC:COMPONENT>
```

Appendix C:

Exemplary HTC File Defining An Exemplary CTS Preview Tag

```
<!--

component name:
    preview

usage:
    <crs:preview id="previewArea" module-id="337" >
        (content inserted by preview element)
        <crs:resource resource-id="336" />
    </crs:preview>

notes:
    - uses the module-id to query the context object for the structural html in the DB. Places this
        html within the preview tag.

-->
<PUBLIC:COMPONENT tagName="preview" literalcontent="false"
    urn="microsoft.com/hotmail/frontdoor/uidev/crs">
    <PUBLIC:ATTACH event="oncontentready" onevent="init()" />
    <PUBLIC:METHOD name="refresh" />
    <SCRIPT language="javascript">
        var m_mid = null;
        function init()
        {
            refresh();
        }
        function refresh()
        {
            //get the module id from the context object
            m_mid = window.document.all.context.moduleTarget;

            //clear the contents first
            element.innerHTML = "";

            //populate the structural html using the context object
            populateStructuralHTML();
        }
        function populateStructuralHTML()
        {
            try
            {
                try
```

```
                {
                        if( isFinite(m_mid) == true && m_mid != null)
                        {
                                var structureHtml =
context.getStructuralHtmlForModule();
                                element.insertAdjacentHTML("afterBegin",
structureHtml);
                        }
                        else
                        {
                                element.insertAdjacentHTML("afterBegin", "(error: invalid
module-id: " + m_mid + ". )");
                        }
                }
                catch(e)
                {
                        throw "problems: \n" + e;
                }
        }
        catch(e)
        {
                alert("Preview failed to initialize. [location: preview.htc:init()]\n" + e);
        }
    }
</SCRIPT>
</PUBLIC:COMPONENT>
```

Appendix D:

Exemplary HTC File Defining An Exemplary Context Object

```
<!--
component name:
    context

usage:
    <crs:context id="context"/>

notes:
    - events fired from this object will have a srcUrn property of the urn specified in the component
    tag attributes.
    - this element does not need to respond to oncontentready since it contains no markup that needs
    to be parsed.
    - requires crs_common.js
    - as a general rule, we never allow clients of this object direct access to the resource cache
    (m_x_cache)
      as a defensive mechanism to ensure this object can count on the integrity of the cache.

dependencies:
    templates
    includes
    stored procedures
    etc

-->
<PUBLIC:COMPONENT tagName="context" lightweight="true" literalcontent="true"
    urn="microsoft.com/hotmail/frontdoor/uidev/crs/context">
    <PUBLIC:DEFAULTS canHaveHTML="false" tabStop="false" contentEditable="false" />
    <PUBLIC:METHOD name="getModuleResources" />
    <PUBLIC:METHOD name="setResourceValueNode" />
    <PUBLIC:METHOD name="getCompiledResourceValue" />
    <PUBLIC:METHOD name="getStructuralHtmlForModule" />
    <PUBLIC:METHOD name="instanceExistsForResource" />
    <PUBLIC:METHOD name="getResourceValueNode" />
    <PUBLIC:METHOD name="zipnav" />
    <PUBLIC:METHOD name="saveChanges" />
    <PUBLIC:METHOD name="readAhead_getModuleValueNode" />
    <PUBLIC:METHOD name="expireReadAhead_getModuleValueNode" />
    <PUBLIC:PROPERTY name="queryTarget" get="get_queryTarget" put="put_queryTarget" />
    <PUBLIC:PROPERTY name="moduleTarget" get="get_moduleTarget"
    put="put_moduleTarget" />
```

```
<PUBLIC:PROPERTY name="releaseTarget" get="get_releaseTarget"
put="put_releaseTarget"/>
<PUBLIC:PROPERTY name="designMode" get="get_designMode" put="put_designMode"/>
<SCRIPT language="javascript" src="inc/crs_common.js"></SCRIPT>
<SCRIPT language="javascript" src="inc/validation.js"></SCRIPT>
<SCRIPT language="javascript">
        //publics
        var m_i_queryTarget = 0;
        var m_i_moduleTarget = 0;
        var m_i_releaseTarget = 0;
        var m_i_user_id = 0;
        var m_e_preview = null;
        var m_x_cache = null;
        var m_x_def_cache = null;
        var m_b_designMode = false;
        //init code
        init();
        //functions
        function init()
        {
                if (get_designMode()==false)
                {
                        //if we are not in design mode, then use session data to determine
environment settings.
                        m_i_queryTarget = get_current_query_id();
                        m_i_moduleTarget = get_current_module_id();
                        m_i_releaseTarget = get_current_release_id();
                        m_i_user_id = get_current_user_id();
                }
                else
                {
                        //design mode on, check for environment data in the querystring
                        try
                        {
                                m_i_queryTarget = window.location.search.match(/query-
target=\d+/)[0].match(/\d+/)[0];
                                m_i_moduleTarget = window.location.search.match(/module-
target=\d+/)[0].match(/\d+/)[0];
                                m_i_releaseTarget = window.location.search.match(/release-
target=\d+/)[0].match(/\d+/)[0];

                                //alert("query: " + m_i_queryTarget + ", module: " +
m_i_moduleTarget + ", release: " + m_i_releaseTarget);  //debug
                        }
                        catch(e)
                        {
                                //[ ] todo: support additional methods to retrieve the params
```

```
                  //                              - possibly via hidden fields on the doc

                        alert("Unable to resolve query-target/module-target/release-target
in querystring. (context)");
                        }
                }
                rebuildCache();
        }
        function get_designMode()
        {
                return m_b_designMode;
        }
        function put_designMode(newMode)
        {
                //alert("newMode: " + newMode);  //debug
                if (newMode == 0)
                {
                        m_b_designMode = false;
                }
                else
                {
                        m_b_designMode = true;
                        init();
                }
        }
        function get_queryTarget()
        {
                return m_i_queryTarget;
        }
        function get_moduleTarget()
        {
                return m_i_moduleTarget;
        }
        function zipnav(release_id, module_id, query_id)
        {
                //no error handling here
                m_i_releaseTarget = release_id;
                m_i_moduleTarget = module_id;
                m_i_queryTarget = query_id;

                //clear the cache and refresh using the new criteria
                rebuildCache();

                return;
        }
        function put_queryTarget(query_id)
        {
```

```
            //set the internal query_id
            m_i_queryTarget = query_id;
            rebuildCache();
}
function put_moduleTarget(module_id)
{
            //set the internal module_id
            m_i_moduleTarget = module_id;
            rebuildCache();
}
function put_releaseTarget(release_id)
{
            //set the internal release_id
            m_i_releaseTarget = release_id;
            rebuildCache();
}
function get_releaseTarget()
{
            return m_i_releaseTarget;
}
function getModuleResources()
{
            //return a pointer to a clone of the module node from the resource cache
            try
            {
                    return m_x_cache.selectSingleNode("//module").cloneNode(true);
            }
            catch(e)
            {
                    //do nothing.
            }
}
function rebuildCache()
{
            /*
            function
                    rebuildCache 1.0
            parameters
                    none
            returns
                    none
            notes
                    - builds the cache using the query and module id in the environment,
overwrites the
                    cache if data already exists in it.
            */
```

```
          //query for module resource values and load into cache
          var url = get_webserver_name() + "templates/resourceValues.asp?query_id=" +
get_queryTarget() +
                                "&module_id=" + get_moduleTarget() + "&release_id=" +
get_releaseTarget() + "&resolveExternals=1";
          //if (window.confirm("cache rebuilt, do you want to view its
contents?")==true){window.open(url);} //debug
          m_x_cache = accessXML(url);
          //set query-target information in cache (used by checkout.xsl)
          m_x_cache.documentElement.setAttribute("query-target", get_queryTarget());
          //set instance ids for those which were referred
          var e = new Enumerator(m_x_cache.selectNodes("//instance[@referred-by]"));
          for (; !e.atEnd(); e.moveNext())
          {
                //set the instance id
                e.item().setAttribute("instance-id", e.item().getAttribute("referred-by"));
          }
}
function setResourceValueNode(resourceID, valueNode, preferredTarget)
{
          /*
          function
                setResourceValueNode 1.0
          parameters
                resourceID (integer) - the id of the resource to set
                valueNode (IXMLDOMNode) - the datatype specific instance element to
set as content
          returns
                true upon success, false on failure
          notes
          */
          var resource_xpath = "//module/resource[@resource-id=" + resourceID + "]";
          var resource_node = m_x_cache.selectSingleNode(resource_xpath);

          //check quality of valueNode parameter
          if (valueNode == null)
          {
                alert("The context element's setResourceValueNode method requires a non
null value for the valueNode parameter.");
                return false;
          }
          if (typeof valueNode != "object")
          {
                alert("You can only pass IXMLDOMNode objects to the
setResourceValueNode method.\nYou passed in a parameter of type: " + typeof valueNode + ".");
                return false;
          }
```

```
                //make sure instance-id is not provided
                if (valueNode.getAttribute("instance-id") != null)
                {
                        alert("You cannot pass instance-id data to the setResourceValueNode
method.\nNode not added.\n(setResourceValueNode (context))");
                        return false;
                }
                //alert("resId: " + resourceID + "\nsetResourceValueNode xml:\n" +
valueNode.xml + "\ntarget resource node: \n\n" + resource_node.xml);  //debug
                if (resource_node != null)
                {
                        //set dirty
                        resource_node.setAttribute("dirty", "true");
                        //create a new instance node if necessary
                        var instance_node = resource_node.selectSingleNode("instance[@query-
id=" + get_queryTarget() + "]");
                        if (instance_node == null)
                        {
                                //unable to find any pre-localized instances, check for ones which
have been referred
                                instance_node =
resource_node.selectSingleNode("instance[@referred-by]");
                                if (instance_node != null)
                                {
                                        //instance node was located, alter the query id so that the
context object can locate it from now on
                                        instance_node.setAttribute("query-id", get_queryTarget());
                                }
                                else
                                {
                                        //instance was not available from a referrer either, create a
new one
                                        var i_node = m_x_cache.createElement("instance");
                                        i_node.setAttribute("resource-id",
resource_node.getAttribute("resource-id"));
                                        i_node.setAttribute("query-id", get_queryTarget());
                                        i_node.setAttribute("release-id", get_releaseTarget());
                                        //if this is a 'dead instance', copy the instance id from the
top level referrer data
                                        var dead_referrer = resource_node.getAttribute("referred-
by");
                                        if (dead_referrer != null)
                                        {
                                                i_node.setAttribute("instance-id", dead_referrer);
                                        }
                                        //add the instance node to the hierarchy
                                        instance_node = resource_node.appendChild(i_node);
```

```
                              }
                    }
                    //set the target-query-id
                    if (preferredTarget)
                              instance_node.setAttribute("target-query-id", preferredTarget);
                    else
                              instance_node.setAttribute("target-query-id", crs_null_keyword);
                    //add the dt node to the cache
                    if (instance_node.hasChildNodes())
                    {
                              //replace children
                              instance_node.replaceChild(valueNode.cloneNode(true),
instance_node.firstChild);
                    }
                    else
                    {

                              //append children
                              instance_node.appendChild(valueNode.cloneNode(true));


                    }
          }
          else
          {

                    alert("Null resourceID passed to the setResourceValueNode
method.\nNode not added.\n(setResourceValueNode (context))");
                    return false;
          }
          //update any resource/if-resource tags referencing this id
          updateResource(resourceID);
          //return good
          return true;
}
function updateResource(resourceID)
{
          /*
          function
                    updateResource 1.0
          parameters
                    resourceID (integer) - the id of the resource to update
          returns
          notes
                    - this function searches the hosting document for any related
crs:resource/crs:if-resource tags
                    and calls the refresh method on them.
          */

          //locate the preview element
```

```
        var p = getPreviewElement();

        //locate resource/if-resource tags within it
        if (p != null)
        {
                //if-resource
                var col_if_resources = p.all.tags("if-resource");
                for (var i=0; i<col_if_resources.length; i++)
                {
                        var elem = col_if_resources[i];
                        if (elem.getAttribute("resource-id") == resourceID)
                        {
                                elem.refresh();
                        }
                }
                //resource
                var col_resources = p.all.tags("resource");
                for (var i=0; i<col_resources.length; i++)
                {
                        var elem = col_resources[i];
                        if (elem.getAttribute("resource-id") == resourceID)
                        {
                                elem.refresh();
                        }
                }
        }

}
function getPreviewElement()
{
        /*
        function
                getPreviewElement 1.0
        parameters
        returns
                the preview element from the page
        notes
                - uses caching
                - locates the preview element anywhere within the document heirarchy
        */

        if (m_e_preview == null)
        {
                //search for it
                var colPreviews = window.document.all.tags("preview");
                if (colPreviews.length > 0)
                {
```

49

```
                              m_e_preview = colPreviews[0];
                              return m_e_preview;

                      }
              }
              else
              {
                      return m_e_preview;
              }
      }
      function saveChanges(fn_validateHandler)
      {
              /*
              function
                      saveChanges 1.0
              parameters
                      [fn_validateHandler] (function pointer) - (optional) string name of the
function which can handle
                      validation errors.
              returns
                      true on success, false on failure
              notes
                      - called by the ui to persist changes in db
              */

              //check design mode -------------------------------------------------------------------
----------\

              if (get_designMode()==true)
              {
                      alert("Changes can not be saved in design mode.");
                      return false;
              }
              //check if this module-query-version is checked out -----------------------------------
---------------\

              if (isModuleCheckedOut(get_releaseTarget(),

                                              get_moduleTarget(),
                                              get_queryTarget())==true)

              {
                      alert("Changes can not be saved because\nthis module is currently
checked out.");
                      return false;
              }
              //check if this module-query-version is signed off -----------------------------------
-------------\

              if (isModuleSignedOff(get_releaseTarget(),

                                              get_moduleTarget(),
                                              get_queryTarget())==true)

              {
```

MS1-1483US

lee&hayes pllc   509-324-9256

```
                    alert("Changes can not be saved because\nthis module is currently signed
off.");
                    return false;
          }
          //verify the current user has write access to the module ------------------------------
-------------\
          if (userHasAccess(m_i_user_id,
                                        get_releaseTarget(),
                                        get_moduleTarget(),
                                        get_queryTarget(),
                                        true,
                                        false,
                                        false,
                                        true)==false)
          {
                    //user does not have access to save the module
                    return false;
          }
          //validate resources ----------------------------------------------------------------
--------\
          if (validation_validateResources(m_x_cache, fn_validateHandler)==false)
          {
                    alert("Validation failed.\nCannot save.");
                    return false;
          }
          //process changes and post query to server -------------------------------------------
------------\
          var xslt = accessXML(get_webserver_name() + "stylesheets/context_update.xsl");
          //transform cache
          var diff = getNewXMLDoc();
          m_x_cache.transformNodeToObject(xslt.documentElement, diff);

          //window.clipboardData.setData("Text",diff.xml);  //debug -- copy updategram to
clipboard
          //if (window.confirm(("Post diffgram?\n\n" + diff.xml))==false){return false;}
//debug

          //post to server
          var x_result = accessXML(get_webserver_name() + "update_handler.asp",
"POST", diff);
          //evaluate quality
          if (updateSucceeded(x_result, true) == false)
          {
                    alert("Update failed.");
                    return false;
          }
          else
```

```
                        {
                                //alert("save result xml: \n\n" + x_result.xml);  //debug
                                alert("Changes saved.");
                                rebuildCache();
                                return true;
                        }
                }
                function readAhead_getModuleValueNode(queryID, noReferrer)
                {
                        /*
                        function
                                readAhead_getModuleValueNode 1.0
                        parameters
                        returns
                        notes
                                - to speed up access to calls to getResourceValueNode
                        */

                        if (!queryID)
                                queryID = get_queryTarget();
                        if (!noReferrer)
                                noReferrer = "0";

                        //get data
                        var url = get_webserver_name() + "templates/resourceValues.asp?query_id=" +
queryID +
                                                        "&module_id=" + get_moduleTarget() + "&release_id=" +
get_releaseTarget() +
                                                        "&no_referrer=" + noReferrer + "&resolveExternals=1";

                        //set cache
                        m_x_def_cache = accessXML(url);
                }
                function expireReadAhead_getModuleValueNode()
                {
                        /*
                        function
                                expireReadAhead_getModuleValueNode 1.0
                        parameters
                        returns
                        notes
                                - expires cache so subsequent calls to getResourceValueNode will return
accurate data.
                        */

                        m_x_def_cache = null;
                }
```

MS1-1483US

```
function getResourceValueNode(resourceID, queryID, noReferrer)
{
        /*
        function
                getResourceValueNode 1.0
        parameters
                resourceID (integer) - the id of the resource to retrieve
                queryID (integer) - the target query id to lookup
                noReferrer (1/[0]) - affects whether referrers are used in the lookup
        returns
                <resource> node complete with instance and dt instance nodes.
        notes
        */

        //fixup parameters
        if (!queryID)
                queryID = get_queryTarget();
        if (!resourceID)
                alert("Null resource id was passed to getResourceValueNode function in
context object.");
        if (!noReferrer)
                noReferrer = "0";

        if (m_x_def_cache)
        {
                //if the read ahead cache is set, read from that instead
                var z = m_x_def_cache.selectSingleNode("//resource[@resource-id=" +
resourceID + "]");

                return m_x_def_cache.selectSingleNode("//resource[@resource-id=" +
resourceID + "]");
        }
        else
        {
                //get data
                var url = get_webserver_name() +
"templates/resourceValues.asp?query_id=" + queryID +
                                                "&resource_id=" + resourceID + "&release_id=" +
get_releaseTarget() +
                                                "&no_referrer=" + noReferrer;
                var x_doc = accessXML(url);

                //return results (no need to use protection via cloning)
                return x_doc.selectSingleNode("//resource");
        }
}
function getCompiledResourceValue(resourceID, crs_element)
{
```

```
/*
function
        getCompiledResourceValue 1.0
parameters
        resourceID (integer) - the id of the resource to compile
        [crs_element] (IHTMLDOMNode) - optional reference to <crs:resource>
element
        returns
        HTML representing resource
        notes
        - this method attempts to best match a resourceNode with an
instanceNode using the query target
                specified, and passes this information to instanceNodeToHTML for
final rendering.
        - externally exposed method
*/


        var resource_node = m_x_cache.selectSingleNode("//resource[@resource-id=" +
resourceID + "]");


        if (resource_node)
        {
                //instance matching logic
                var instance_node =
getBestInstanceNodeFromResourceNode(resource_node);
                //render the resource and instance
                return instanceNodeToHTML(resource_node, instance_node,
crs_element);
        }
        else
        {
                //the resource is not present in the cache, add it
                resource_node = addResourceToCache(resourceID);
                if (resource_node)
                {
                        //instance matching logic
                        var instance_node =
getBestInstanceNodeFromResourceNode(resource_node);
                        //render the resource and instance
                        return instanceNodeToHTML(resource_node, instance_node,
crs_element);
                }
                else
                {
                        //resource can not be found
                        return "";
                }
```

```
            }
    }
    function instanceExistsForResource(resource_id)
    {
            /*
            function
                    instanceExistsForResource 1.0
            parameters
                    resource_id (int) - resource id
            returns
                    (bool) true/false indicating whether an instance exists for the resource
within
                    the cache.
            notes
                    - this function is externally exposed as a method
                  . - intended to be called by the if-resource element
                    - this object also counts on the fact that an instance node will never exist
without
                    a matching dt-specific instance row, which should be maintained by logic
within the db.
            */

            var i_node = m_x_cache.selectSingleNode("//instance[../@resource-id=" +
resource_id + "]");

            if (i_node != null)
                    return true;
            else
                    return false;
    }
    function getBestInstanceNodeFromResourceNode(resourceNode)
    {
            /*
            function
                    getBestInstanceNodeFromResourceNode 1.0
            parameters
                    resourceNode (IXMLDOMNode) - the resource node used to query for
instance data
            returns
                    an instance node, or null
            notes
                    - this method attempts to locate the most specific instance node from a
resource
                    node, based on the query target set in the context object's properties.
                    - intended to be used by the getCompiledResourceValue method
            */
```

```
                    var instance_node = resourceNode.selectSingleNode("instance[@query-id=" +
get_queryTarget() + "]");

                if (instance_node == null)
                {
                        //try to locate any instance
                        instance_node = resourceNode.selectSingleNode("instance");
                }

                return instance_node;
        }
        function addResourceToCache(resourceID)
        {
                /*
                function
                        addResourceToCache 1.0
                parameters
                        resourceID (integer) - the id of the resource to add to the cache
                returns
                        the resource node added to the cache (IXMLDOMNode), or null.
                notes
                        - this function adds a resource that is normally outside the module target
selected.
                        - intended to be used from structural preview html that tries to render a
resource that is
                        not already present in the cache since it belongs to a module outside of
the current one.
                        - calls to this method make full use of the defaulting system, including
referrers
                */

                var url = get_webserver_name() + "templates/resourceValues.asp?query_id=" +
get_queryTarget() +
                                        "&resource_id=" + resourceID + "&release_id=" +
get_releaseTarget();

                var x_data = accessXML(url);
                var resource_node = x_data.selectSingleNode("//resource");

                if (resource_node == null)
                {
                        if (get_designMode()==false)
                        {
                                //resource could not be found in the database
                                alert("Resource id: " + resourceID + " could not be found in the
database.");
                                return null;
```

```
                    }
                    else
                    {
                            //user is using the context.htc from within the context of a
structural editor
                            return null;
                    }
            }
            else
            {
                    //move the resource node to the cache
                    return
m_x_cache.selectSingleNode("//root").appendChild(resource_node);
            }
     }
     function getExtendedAttributeList(crs_element)
     {
            /*
            function
                    getExtendedAttributeList 1.0
            parameters
                    [crs_element] (IHTMLDOMNode) - optional reference to <crs:resource>
node
            returns
                    string representing attribute list
            notes
                    this function examines attributes expressed on a <crs:resource> node, and
builds a name/value pair string
                    so those attributes can be passed down to an element such as a <a> or
<img>.
            */

            var extra_attributes = "";

            if (crs_element)
            {
                    //process standard attributes
                    for (var i=0; i<crs_element.attributes.length; i++)
                    {
                            var attrib = crs_element.attributes[i];
                            switch (attrib.nodeName)
                            {
                                    case "resource-id":
                                            {
                                                    //do nothing
                                            }
                                    break;
```

57

```
                        default:
                                {
                                        if (attrib.nodeValue)
                                        {
                                                if (attrib.specified)
                                                {
                                                        extra_attributes += " " +
attrib.nodeName + "=\"" + attrib.nodeValue + "\"";
                                                }
                                        }
                                }
                                break;
                        }
                }
                //process style attribute (special case)
                var h = crs_element.outerHTML.toLowerCase();
                var s_loc = h.indexOf("style=\"");
                if (s_loc > 0)
                {
                        s_loc += 7;  //length of string: style="
                        var s_eloc = h.indexOf("\"", s_loc);
                        var s_style = h.substr(s_loc, s_eloc - s_loc);
                        extra_attributes += " style=\"" + s_style + "\"";
                }
        }

        return extra_attributes;
}
function instanceNodeToHTML(resourceNode, instanceNode, crs_element)
{
        /*
        function
                instanceNodeToHTML 1.0
        parameters
                resourceNode (IXMLDOMNode) - the resource node
                instanceNode (IXMLDOMNode) - the instance node which contains a
child of the datatype specific

instance node, or no children, or NULL
                [crs_element_attributes] (IHTMLDOMNode/Collection) - optional
collection of attributes from a <crs:resource> element
                returns
                        HTML representing instance
                notes
                        - both the resource and instance nodes are passed to this method so that
for datatypes that do
```

58

not require instances eg. cleargif, can still be rendered, as well as for datatypes that

require resource data to be rendered properly.

```
*/
var type = resourceNode.selectSingleNode("type").getAttribute("type-name");
var res = "";
var dtNode;

if (instanceNode != null)
{
        //locate the datatype specific instance node if available
        if (instanceNode.hasChildNodes())
        {
                dtNode = instanceNode.firstChild;
                switch (type) //only support types that have a visual representation
in a structural preview
                {
                        case "text":
                                res = nz(dtNode.getAttribute("text"));
                                break;
                        case "link":
                                if (nz(dtNode.getAttribute("url")).toString().length
> 0 || nz(dtNode.getAttribute("text")).toString().length > 0)
                                {
                                        var extra_attributes =
getExtendedAttributeList(crs_element);

                                        var url_part =
nz(dtNode.getAttribute("url"));


                                        //resolve rp node attributes
                                        var s_i_url_only = "0";
                                        try
                                        {
                                                s_i_url_only =
resourceNode.firstChild.nextSibling.getAttribute("url-only");
                                        }
                                        catch(e){}

                                        //build output
                                        if (s_i_url_only == "1")
                                        {
                                                //url only link
                                                res = url_part;
                                        }
                                        else
                                        {
                                                //normal link
```

```
                                                                    res = "<a target=\"_new\" href=\"" +
url_part + "\"" + extra_attributes + ">" + nz(dtNode.getAttribute("text")) + "</a>";
                                                              }

                                                    }
                                                    break;
                                          case "image":
                                                    {
                                                              var image_data_id =
parseInt(dtNode.getAttribute("image-data-id"),10);
                                                              if (image_data_id > 0)
                                                              {
                                                                        var extra_attributes =
getExtendedAttributeList(crs_element);
                                                                        var url = get_webserver_name() +
"/crsdata-anon/db/image_data[@image_data_id=" + image_data_id + "]/@image_data";
                                                                        var img = "<img border=0 src=\"" +
url + "\" alt=\"" + nz(dtNode.getAttribute("title")) + "\" " + extra_attributes + " >";
                                                                        if
(dtNode.getAttribute("url").toString().length > 0)

                                                                        {
                                                                                  res = "<a target=\"_new\"
href=\"" + dtNode.getAttribute("url") + "\">" + img + "</a>";  //link specified
                                                                        }
                                                                        else
                                                                        {
                                                                                  res = img;  //link not
specified
                                                                        }
                                                              }
                                                    }
                                                    break;
                                        }
                              }
                    }
                    else
                    {
                              if (get_designMode() == true)
                              {
                                        res = getDeisnModeDefaultHTMLForResource(resourceNode);
                              }
                    }
                    //return value
                    return res;
          }
          function getDeisnModeDefaultHTMLForResource(resourceNode)
          {
```

60

```
            return "(resource-id: " + resourceNode.getAttribute("resource-id") + ", " +
resourceNode.getAttribute("short-name") + ")";
        }
        function getStructuralHtmlForModule()
        {
            /*
            function
                    getStructuralHtmlForModule 1.0
            parameters
                    none.
            returns
                    HTML representing structural html.
            notes
                    - uses the module target of the context object to locate the module.
            */

            var module_node = m_x_cache.selectSingleNode("//module[@module-id=" +
get_moduleTarget() + "]");

            if (module_node != null)
            {
                //prefer the structural resource node which is not for output
                var structural_resource_node = null;
                structural_resource_node =
module_node.selectSingleNode("resource[type/@type-name='structure' and structure-resource-
properties/@for-output = '0']");

                if (!structural_resource_node)
                {
                    //unable to locate a preview specific html, try a non preview
specific one
                    structural_resource_node =
module_node.selectSingleNode("resource[type/@type-name='structure' and structure-resource-
properties/@for-output = '1']");
                }

                //get data
                if (structural_resource_node)
                {
                    var structural_instance_node =
structural_resource_node.selectSingleNode("instance/structure-instance");
                    if (structural_instance_node != null)
                    {
                        return structural_instance_node.getAttribute("html");
                    }
                    else
                    {
```

61

```
                                //no instance node found
                                //window.status = "no instance node found";  //debug
                                return "";
                        }
                }
                else
                {
                        //window.status = "no resource node found";  //debug
                        //no resource node found
                        return "";

                }
        }
        else
        {
                //window.status = "no module node found";  //debug
                //no module node found
                return "";

        }
    }
</SCRIPT>
</PUBLIC:COMPONENT>
```